UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Desarrollo de un juego serio multiplataforma para introducir la lectura a niños con TEA

Sergio Martínez Carrasco

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de un juego serio multiplataforma para introducir la lectura a niños con TEA

Autor: Sergio Martínez Carrasco
Tutor: Germán Montoro Manrique

julio 2020

Sergio Martínez Carrasco
Desarrollo de un juego serio multiplataforma para introducir la lectura a niños con TEA

Sergio Martínez Carrasco
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos

Agradecimientos

Me gustaría comenzar agradeciendo a las personas que me han dado la oportunidad para poder estudiar lo que me apasiona, mis padres, Luis Javier Martínez y María Dolores Carrasco que sin su apoyo y sin su ayuda no hubiera podido llegar a donde estoy. Quiero agradecer también todo lo que me han enseñado y todo lo que gracias a ellos ha hecho que sea la persona que soy hoy. También quiero darle las gracias a mi hermana Nuria, una gran amiga y una persona muy importante para mí que siempre ha estado cuando la he necesitado. Sin olvidarme de mi cuñado Jorge, por su nobleza y por su apoyo.

En general a toda mi familia que me cuida, me ha apoyado y ha creído en mí.

Gracias también a mi tutor Germán Montoro por su ayuda y apoyo a lo largo de este trabajo y por darme la oportunidad de trabajar en un proyecto tan enriquecedor.

También me gustaría mencionar a los que han sido mis compañeros durante la carrera y, al final de ella, amigos. A Álvaro Sosa por estar ahí conmigo en todas las batallas, Álvaro Caseiro por la ayuda y los consejos, a Lucas Ruiz por soportarme también en los momentos no tan buenos, a todos ellos y a Víctor, Ricardo, Daniel, Sergio por las risas y por hacer mejor la experiencia en la universidad. Gracias.

Resumen

La lectura requiere el desarrollo de habilidades como el reconocimiento de una palabra individualmente hasta entender el contexto en el que se usa. Es importante manejar bien estas habilidades ya que la lectura proporciona independencia, es un medio que utilizamos para comunicarnos y una herramienta fundamental para desarrollarnos y relacionarnos. Las personas con Trastornos del Espectro Autista (TEA) encuentran difultades a la hora de aprender a leer utilizando los métodos silábicos tradicionales. Para favorecer que estos niños tengan esa independecia, no tengan que sufrir el aislamiento social que produce la falta de comunicación, es necesario buscar otros métodos con los que enseñar algo tan fundamental como la lectura.

Gracias a las grandes capacidades visuales que poseen los niños con TEA, el aprendizaje basado en el métodos de lectura global se ha convertido en la mejor forma de introducir a estos niños en la lectoescritura. Debido a que pueden diferenciar imágenes y asociarlas con su signficado de forma sencilla, se ha trabajado con éste método mediante pictogramas físicos hechos a mano por personal docente. Este trabajo se ha facilitado con la introducción de la tecnología, que nos permite disponer de estas imágenes en cualquier lugar y en cualquier momento.

Actualmente resulta complicado encontrar aplicaciones que trabajen la lectura global y más aún aplicaciones enfocadas al desarrollo de la lectura en personas con TEA. Leo con Lula es una de las aplicaciones que cumple ambos requisitos y actualmente existe una versión para dispositivos iOS. La importancia de iniciar a estos niños con capacidades especiales en la lectura hace que sea necesario ampliar el número de personas a las que pueda llegar esta tecnología. El objetivo de este TFG es diseñar y desarrollar una aplicación multiplataforma funcional basada en la actual versión para iOS de Leo con Lula que permita mantener el objetivo principal de ayudar a iniciar a niños con TEA en la lectura consiguiendo llegar a más dispositivos.

A lo largo del documento se va a describir el proceso que se ha seguido para cumplir con los objetivos planteados. Este proceso comienza con el análisis de los requisitos, el diseño planteado y las decisiones que se han tomado para su implementación.

Palabras clave

Lectoescritura, multiplataforma, Trastornos del Espectro Autista, lectura global, pictograma, juego serio, TIC.

Abstract

Reading requires the development of skills such as recognition of an individual word until the context in which it is used is understood. It is important to manage these skills well because reading provides independence, is a means we use to communicate and a fundamental tool for developing and relating. People with Autism Spectrum Disorders (ASD) find it difficult to learn to read using traditional syllabic methods. To encourage these children to have this independence, so that they do not have to suffer the social isolation that a lack of communication produces, it is necessary to look for other methods with which to teach something as fundamental as reading.

Thanks to the great visual abilities that children with ASD have, learning based on global reading methods has become the best way to introduce these children to reading and writing. Because they can differentiate images and associate them with their meaning in a simple way, this method has been worked with by means of physical pictograms handmade by teaching staff. This work has been facilitated by the introduction of technology, which allows us to have these images available anywhere and at any time.

Nowadays it is difficult to find applications that work with global reading and even more so applications focused on the development of reading skills in people with ASD. 'Leo con Lula' is one of the applications that meets both requirements and there is currently a version for iOS devices. The importance of initiating these children with special abilities in reading makes it necessary to expand the number of people this technology can reach. The objective of this final degree project is to design and develop a functional cross-platform application based on the current iOS version of Leo con Lula that will allow to maintain the main objective of helping to initiate children with ASD in reading by reaching more devices.

Throughout the document we are going to describe the process that has been followed in order to fulfil the objectives set out. This process starts with the analysis of the requirements, the application design and the decisions that have been taken for its implementation.

Keywords

Reading and writing, cross-platform, Autism Spectrum Disorders, global reading, pictogram, serious game, ICT.

Índice

Listas

Lista de códigos

Lista de figuras

1

Introducción

1.1. Motivación

Desde que somos pequeños aprendemos a leer y escribir utilizando el método silábico. Este consiste en aprender inicialmente las letras del abecedario y según se van asimilando se aprende a combinar las sílabas y cómo es su sonido. Con este método, niños con Trastornos del Espectro Autista (TEA), muestran más dificultades a la hora de aprender a leer. Estos niños presentan perfiles de lectura caracterizados por habilidades de decodifación más altas y menor comprensión lectora por lo que no son capaces de extraer el significado de la palabra. Pero, aunque puedan tener ese tipo de limitación, estos niños tienen habilidades visuales superiores que hacen que el sistema de lectura global sea el mejor para el aprendizaje de la lectoescritura.

Con este propósito se crea Leo con Lula [1]. Se trata de una aplicación para dispositvos móviles que también cuenta con una versión para pizarra digital y que contiene ejercicios basados en el método de lectura global cuyo objetivo es el de despertar el interés del niño por la lectura.

La lectura global funciona de manera inversa a la lectura silábica tradicional ya que comienza directamente asociando la imagen con su palabra escrita y su sonido. A continuación se realiza la descomposición silábica con el sonido de las mismas y por último se introduce las letras y su fonética.

Actualmente para el proyecto Leo con Lula existe un desarrollo en iOS [2] que está disponible en la Apple Store pero es necesario que esté disponible en más dispositivos y tipos de dispositivos. En el pasado se desarrollaron versiones nativas en Android de Leo con Lula pero ahora mismo estas versiones están obsoletas. Por ese motivo se plantea el desarrollo utilizando una herramienta multiplataforma. En el desarrollo de software mantener una única versión para diversas plataformas y dispositivos es algo que se valora, puesto que reduces costes de mantenimiento del código y aumentas el alcance de tu apliación dándote la posibilidad de llegar a más gente. Teniendo en cuenta cómo el proyecto de Leo con Lula trata de ayudar en el aprendizaje de la lectoescritura de niños con TEA, es más importante aún facilitar el acceso al mayor número de personas.

1.2. Objetivos

El objetivo de este TFG es desarrollar una herramienta software multiplataforma basada en la versión existente de Leo con Lula para dispositivos iOS que permita seguir ayudando en la introducción a la lectura a personas con TEA utilizando el método de lectura global.

Además, se realizarán mejoras sobre la versión ya existente. Se quiere mejorar la gestión de los niveles dentro de la aplicación y que la arquitectura software sea reutilizable y modular.

Se planteará un nuevo diseño de la base de datos que cumpla con las necesidades para la mejora del rendimiento de la aplicación.

También se quiere implementar un nuevo sistema para la gestión del vocabulario que facilite la interacción del usuario y la haga más amigable.

Se pretende añadir cambios en la navegabilidad de la aplicación que hagan que mejore la experiencia de los usuarios, en concreto se quiere facilitar el acceso al menú de gestión del vocabulario y mejorar el método que genera cada uno de los niveles de la aplicación.

1.3. Organización de la memoria

Ésta memoria se divide en los siguientes capítulos:

- **Capítulo 1. Introducción:** motivación y objetivos del proyecto.
- **Capítulo 2. Trabajo relacionado:** En este capítulo voy a hablar sobre el trabajo relacionado con este TFG tanto a nivel de aplicaciones del ámbito de Leo con Lula como diferentes entornos de desarrollo multiplataforma.
- **Capítulo 3. Análisis y diseño:** En este capítulo se va a hablar sobre el análisis de los requisitos de la aplicación así como el diseño definido para el proyecto.
- **Capítulo 4. Implementación:** En este capítulo se van a presentar las decisiones tomadas para implementar las funcionalidades del capítulo 3.
- **Capítulo 5. Conclusiones y trabajo futuro:** Por último, se va a hablar sobre las conclusiones a las que se han llegado en este trabajo y los planes de futuro del mismo.

2

Trabajo relacionado

2.1. Estado del arte

Actualmente no se encuentran muchas herramientas para que niños y niñas con diferentes capacidades puedan aprender a leer o les sirva como un primer acercamiento a la lectura. Sí es cierto que muchos profesores utilizan la lectura global como método de aprendizaje, valiéndose de pictogramas impresos o hechos a mano para que los niños interactúen con ellos y los asocien con palabras.

Esta tarea puede simplificarse considerablemente mediante el uso de las TIC en las aulas. Trasladando los conceptos básicos de la lectura global a una aplicación para un dispositvo móvil se consiguen mejoras en cuanto a la personalización individual de cada alumno y en el seguimiento que se pueda hacer sobre el progreso conseguido.

En este apartado vamos a ver las aplicaciones que existen en la actualidad y que ofrecen la posibilidad de implementar la lectura global como forma de aprendizaje mediante el uso de pictogramas y sonidos.

2.1.1. Leo con Grin

'Leo con Grin' [3] es una iniciativa de Educaplanet para ayudar a los niños en el aprendizaje de la lectura de palabras y la identifiación de sílabas y letras. Mediante una serie de juegos, los niños aprenden a leer mientras se divierten.

Está dividido en dos aplicaciones. 'Leo con Grin' es la primera y consta de 30 lecciones centrándose en las sílabas directamente. La segunda aplicación, 'Aprende a leer 2' consta de 25 lecciones que se basan más en las sílabas inversas, trabadas y mejorar la comprensión lectora.

Cada una de las lecciones cuenta con 13 juegos diferentes y cada uno de ellos cuenta con dos dificultades en función del progreso de los niños.

Figura 2.1: Uno de los juegos disponibles en 'Leo con Grin'.

2.1.2. Yo también leo

'Yo también leo' [4] es una aplicación disponible tanto en Android como en iOS que, como también hace 'Leo con Lula', utiliza el método de lectura global para que niños y niñas con capacidades especiales aprendan a leer.

El juego cuenta con tres niveles siempre disponibles para los niños. Cada nivel cuenta con tres actividades con puzles y juegos de memoria que ayudan a los niños a familiarizarse con el vocabulario.

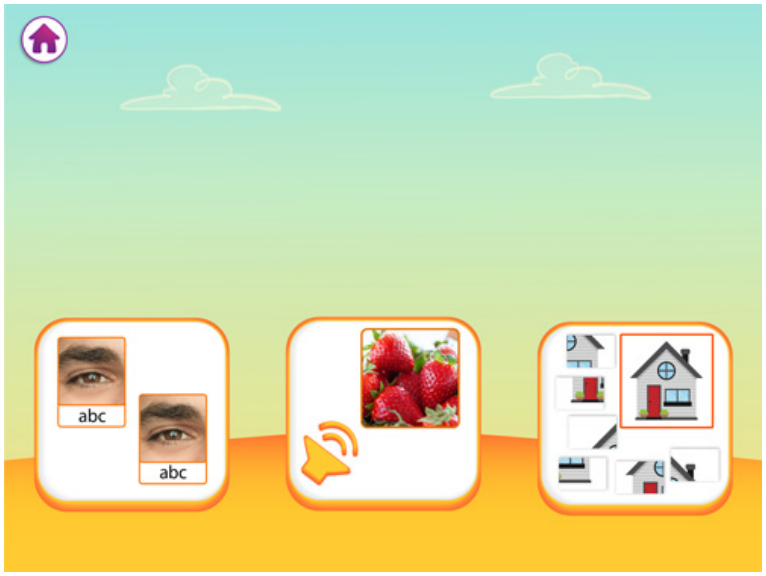

Figura 2.2: Pantalla de selección de nivel de 'Yo también leo'.

Además, los niveles han sido diseñados siguiendo pautas pedagógicas especiales para niños con síndrome de Down, TEA y otros tipos de discapacidad intelectual. Estas pautas incluyen características como la disposición de las actividades de izquierda a derecha y de arriba a abajo para favorecer el

orden real de la lectura, las palabras aparecen con colores diferentes según su función gramatical y todos los niveles combian texto con audio para favorecer al lenguaje de los niños y niñas.

Esta aplicación, además, incluye juegos de recompensa para motivar al niño o la niña.

2.1.3. GloRead

GloRead [5] es una aplicación que trata de enseñar a leer con el método de lectura global. Esta aplicación está disponible en Android de forma gratuita y consta de diferentes modos de aprendizaje.

En cuanto a la configuración, los lenguajes soportados son inglés, hebreo, ruso, alemán, italiano y ucraniano.

El juego incluye 150 palabras, 60 frases, el alfabeto completo y 25 sílabas. En cuanto a los modos de aprendizaje, como se puede observar en la figura 2.3(a) se puede ver que la aplicación nos permite trabajar palabras, sílabas, frases y letras. El juego comienza mostrando los elementos que se van a practicar asociados a una imagen con la posibilidad de reproducir el sonido ya sea de la frase, palabra, sílaba o letra, como se puede apreciar en la figura 2.3(b) para la palabra 'Cow'. Una vez completada la fase de aprendizaje y familiarización con los elementos que se van a trabajar, comenzaría la fase de entrenamiento. Ésta es una fase discriminatoria en la que se muestran cuatro imágenes y el correspondiente elemento de los ya mencioinados. Cuando tocas una de las imágenes la aplicación realizará una evaluación y, en caso de acierto se leerá el elemento, la palabra en el caso de la figura 2.3(c) y se añadirá un 'helado' al marcador que es la forma de recompensa que ofrce GloRead.

(a) Selección de nivel. (b) Aprendizaje de palabra. (c) Discriminación.

Figura 2.3: En estas tres figuras se muestran los procesos principales de la aplicación GloRead.

2.1.4. Vocales y números para niños

Vocales y números [6] se trata de una aplicación educativa que quiere acercar las vocales a los niños con TEA e iniciarlos así en el proceso de la lectoescritura. La aplicación cuenta con apoyo auditivo para todas las letras además de un modo pizarra en la que los niños pueden practicar la escritura de las vocales.

Figura 2.4: Ventana principal de la aplicación.

Figura 2.5: Ejemplo del modo pizarra para números.

La aplicación es gratuíta y únicamente la podemos encontrar en la AppStore en su versión para iOS. Solo dispone de versión en español.

2.1.5. PROLEXyCO

PROLÉXyCO [7] es una aplicación para el desarrollo del lenguaje expresivo y comprensivo. Aunque también es útil para el trabajo con alumnos sin necesidades educativas especiales por servir como

material de apoyo a la lectoescritura, está orientado hacia a alumnos que sí tienen necesidades educativas especiales con afectación en el área del lenguaje.

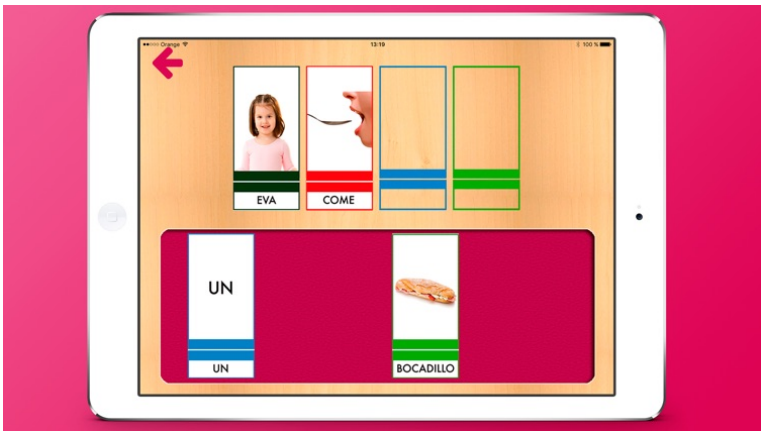

Figura 2.6: Ejemplo de ejercicio que podemos encontrar en la aplicación.

Tan solo disponible para iPhone y iPad es una aplicación gratuita lanzada en el año 2017 y que no ha recibido actualizaciones recientemente.

2.1.6. Conclusiones

Después de buscar información de todas las aplicaciones mencionadas y probar algunas de ellas, se ha comprobado que no hay muchas aplicaciones que utilicen el método de lectura global para el aprendizaje. Algunas de las aplicaciones que sí lo hacen, no están disponibles para todos los dispositivos o bien están obsoletas.

'Leo con Lula' en su versión para iOS se ha convertido en una herramienta muy útil para docentes y padres de niños con capacidades especiales. El hecho de realizar una versión multiplataforma de esta aplicación va a permitr llegar a mucha más gente aprovechando la gran cantidad de dispositivos Android que podemos encontrar en el mercado.

2.2. Estado de la tecnología

El desarrollo de aplicaciones multiplataforma asegura un mismo código para diferentes plataformas. Esto supone una reducción del esfuerzo a la hora de desarrollarlo, no necesitas conocer el código nativo de cada una de las plataformas a desarrollar y tan solo se tiene que mantener un versión del código, lo que reduce considerablemente su coste. Además, cuantas más plataformas soporten la aplicación, mayor es su accesibilidad. Esto último es de gran importancia cuando nos centramos en la aplicación que se desarrolla en este trabajo, ya que sirve de ayuda tanto a profesores, padres como alumnos con TEA.

En esta parte se van a agrupar algunos de los entornos de desarrollo multiplataforma que se han consdierado para la realización de la aplicación.

2.2.1. Unity 3D

Unity [8], desarrollado por Unity Technologies, es un motor de juegos multiplataforma que soporta más de 25 plataformas para móviles, televisiones, aplicaciones para ordenadores, consolas, aplicaciones web, realidad virtual, etc.

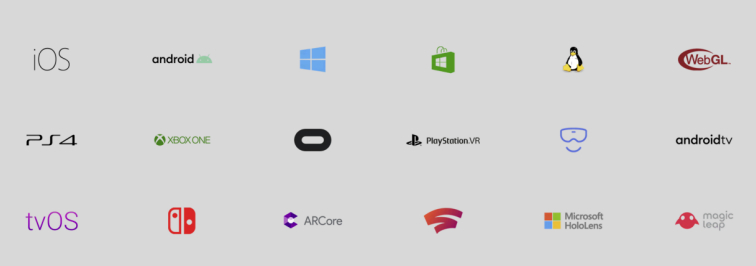

Figura 2.7: Algunas de las plataformas soportadas por Unity 3D

Unity trabaja con C# y para que éste pueda funcionar tanto en Android como en iOS es necesario hacer una conversión ya que Android usa Java o Kotlin para sus aplicaciones e iOS usa Objective-c. Es importante saber cómo Unity 3D realiza esta conversión.

Mono es una implementación de código abierto de .NET Framework de Microsoft basada en los estándares ECMA y CLR, el componente de máquina virtual de Microsoft.

2.2.2. React Native

React Native [9] utiliza JavaScript como lenguaje de programación para crear aplicaciones nativas. Uno de los puntos fuertes de React Native es la posibilidad de crear diferentes módulos en lenguajes como C, Swift o Java.

React Native es, en líneas generales, una de los mejores frameworks a la hora de desarrollar aplicaciones multiplataforma por la velocidad en la que interpreta el código fuente para convertirlo en los elementos nativos de cada plataforma.

Como se muestra en la figura 2.8, React Native se usa para desarrollar aplicaciones para iOS y Android y además para Universal Windows Platform (UWP).

Como punto negativo, React Native no es un framework multiplataforma completo ya que para algunas funcionalidades como la cámara o el acelerómetro es necesario utilizar código separado para Android y para iOS debido a que necesitas usar componentas nativos.

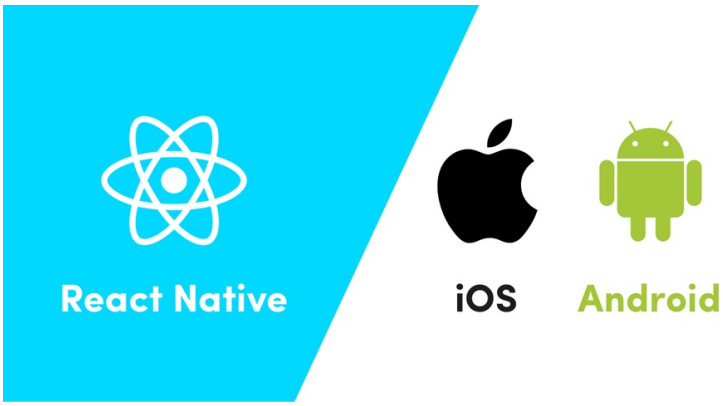

Figura 2.8: Plataformas soportadas por React Native

Algunas de las aplicaciones creadas con el framework multiplataforma de React Native son Pinterest, Skype, Instagram y Tesla.

2.2.3. Ionic

Ionic [10] fué construido a partir del framwork de Angular y Apache Cordova. Como principales teconologías para el desarrollo de aplicaciones, utiliza HTML5, CSS y Javascript. Ionic cuenta con numerosos plugins que pueden ser integrados con los componentes hardware de los dispositivos.

Además, Ionic dispone de un librería extensa de componentes para la interfaz de usuario que imita de manera muy eficiente las interfaces de usuario nativas de cada aplicación.

Cualquier desarrollo en Ionic depede fuertemente de los plugins y, aunque se ha comentado la extensa lista de plugins que posee, es cierto que en ocasiones no es posible encontrar un plugin para controlar componentes hardware muy específicos y tengas que invertir tiempo en el desarrollo del mismo. Además, otro punto negativo es el rendimiento, ya que algunos elementos gráficos en 3D pueden hacer que tu aplicación se relentice por la forma en la Ionic renderiza los elementos gráficos.

2.2.4. Xamarin

Microsoft Visual Studio Xamarin [11] utiliza la plataforma .NET para la cración de aplicaciones para Windows, iOS y Android. Uno de los puntos fuertes de Xamarin es que todas las aplicaciones que se desarrollan con esta herramienta van a ser prácticamente nativas para cada plataforma.

Cuando se habla de desarrollo de juegos, Xamarin te permite utilizar APIs nativas como OpenGL o Apple SceneKit o bien utilizar alguno de los motores de juegos como FlatRedBall [1], Stride [2], Wave

[1] http://flatredball.com/ (08/07/2020)
[2] https://stride3d.net/ (08/07/2020)

Engine [3] o MonoGame [4].

Utilizando Visual Studio App Center puedes monitorizar todo lo relacionado con la aplicación, ya sean errores que fuerzan su cierre, dispositivos en los que se utiliza y toda actividad de usuarios.

Aunque no todo es perfecto ya que Xamarin presenta limitaciones tanto en iOS como en Android a las que no da soporte de forma oficial. Además ciertos elementos gráficos no son compartidos entre las diferentes plataformas haciendo que dichos elementos deban ser codificados en el lenguaje nativo para hacerlos funcionar.

Otro de los puntos negativos es que a la hora de desarrollar juegos para móviles Xamarin no es la primera opción. Debido a la interacción del usuario con elementos gráficos hacen que se pierda el significado de código compartido y se pierde en productividad. Para estos casos es incluso más recomendable realizar dichos juegos en código nativo.

2.2.5. Conclusiones

A pesar de que React Native es uno de los frameworks más utilizado a la hora de crear aplicaciones multiplataforma, el hecho de tener que utilizar compomentes nativos y tener diferentes partes de código para cada plataforma es algo que ha pesado en la decisión de descartarlo.

Lo que hizo plantearme el desarrollo en Ionic fue la disponibilidad de una librearía con componentes de interfaz de usuario ya que 'Leo con Lula' está muy basada en dichos compomentes, es cierto que el rendimiento para objetos 3D es algo que se ha tenido en cuenta.

Por otra parte, Xamarin cuenta con diferentes opciones a la hora de trabajar con motores gráficos y permite utilizar APIs nativas. El problema es que Xamarin no es una opción recomendable para juegos de móvil que requieran interacción con elementos gráficos por parte del usuario.

Aunque Unity no es una de las principales herramientas para el desarrollo de aplicaciones móviles, es cierto que destaca la facilidad con la que se generan versiones para las diferentes plataformas soportadas. Además, Unity cuenta con Unity Asset Store [5] en la que se pueden econtrar multitud de plugins, modelos 3D y 2D gratuitos con acceso directo desde el propio editor de Unity.

Tras valorar las opciones disponibles, cada punto positivo y negativo ellas, he decidido elegir Unity como herramienta para el desarrollo de la versión multiplataforma de Leo con Lula. En los siguientes capítulos se va a tratar cómo va a ser este desarrollo en Unity.

[3] https://waveengine.net/ (08/07/2020)

[4] https://docs.microsoft.com/es-es/xamarin/graphics-games/monogame/?WT.mc_id=docs-dotnet-xamarin (08/07/2020)

[5] https://assetstore.unity.com/ (08/07/2020)

3 Análisis y diseño

En esta parte del documento se van a enumerar los requisitos necesarios desarrollados en la fase de análisis de la aplicación, contemplando los requisitos ya existentes y añadiendo algunas de las mejoras planteadas.

3.1. Análisis de requisitos

En esta sección se recogen los requisitos que surgieron en la fase de desarrollo del proyecto. Aquí se contemplan tanto los requisitos existentes en la versión actual de Leo con Lula como las correcciones y mejoras que se quieren implentar en esta versión.

3.1.1. Requisitos funcionales

Un requisito funcional describe el servicio que el software debe prestar. Puede describir la funcionalidad del sistema o de sus componentes. Se define mediante funciones que se componen de un conjuto de entradas al sistema, el comportamiento establecido para dicha función y sus diferentes salidas.

RF1 Crear usuarios.

- Se debe crear un usuario para poder acceder al resto de funcionalidad de la aplicación.
- Al iniciar la aplicación por primera vez o si no hay usuarios registrados, se solicitará la creación de un nuevo usuario.
- Se podrán crear usuarios desde el menú principal de la aplicación pulsando el bóton de añadir un nuevo usuario. En este momento aparecerá un campo de texto para insertar el nombre del nuevo usuario y un botón para confirmar la acción.
- En caso de que el nombre de usuario ya exista, aparecerá un mensaje de error y se solicitará un nuevo nombre.

RF2 Eliminar usuarios.

- Se podrá eliminar un usuario en cualquier momento desde el menú principal, manteniendo pulsado el icono del usuario que se quiera eliminar. También se podrá eliminar usuarios desde el menú de configuración pulsando

sobre el icono correspondiente.

- Se producirá un error si el usuario eliminado es el único existente en la aplicación puesto que debe haber como mínimo un usuario en todo momento.
- Si se elimina el usuario seleccionado, se seleccionará automáticamente el primer usuario de la lista.

RF3 Seleccionar un usuario.

- Se podrá cambiar de usuario de entre los usuarios que se muestran en el menú principal.
- Para cambiar de usuario es necesario hacerlo desde la pantalla de inicio, pulsando sobre el icono del usuario al que se desea cambiar. La lista con los iconos de los usuarios está situada en la esquina superior izquierda como se puede ver en la figura 3.2.
- Para poder cambiar de usuario es necesario tener al menos dos usuarios creados.

RF4 Cambiar tipo de fuente.

- El usuario podrá cambiar la fuente en la que se muestran las palabras en los diferentes niveles del juego.
- Para poder cambiar la fuente, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando entre Helvética o Chalkboard en el menú que se muestra al pulsarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF5 Activar o desactivar la lectura de palabras.

- El usuario podrá elegir si quiere lectura de palabras durante el desarrollo de los niveles.
- Para poder activar o desactivar la lectura de palabras, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición de leer palabras en el menú que se muestra al pulsarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF6 Establecer el número de aciertos tras los que se elimina el apoyo de palabra.

- El usuario podrá elegir el número de aciertos tras los que desaparece el apoyo de la palabra.
- Para poder establecer este valor, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y e insertando en el campo de texto el número de aciertos necesario para que desaparezca el apoyo de la palabra.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF7 Cambiar mayúsculas y minúsculas.

- Se podrá cambiar el tipo de letra en el que se muestran las palabras en los diferentes niveles del juego.
- Para poder cambiar el tipo de letra, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando entre mayúsculas y minúsculas en el menú que se muestra al pulsarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF8 Cambiar idioma del vocabulario.

- El usuario podrá elegir el idioma de las palabras a la hora de buscar en ARASAAC.
- Para poder activar o desactivar la lectura de palabras, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición de leer palabras en el menú que se muestra al

pulsarlo.

- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF9 Activar o desactivar el sonido en el acierto.

- El usuario podrá elegir si quiere activar o desactiar el sonido cuando se acierte una palabra.
- Para poder activar o desactivar el sonido en el acierto, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición entre activar el sonido o desactivarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF10 Activar o desactivar el sonido en el error.

- El usuario podrá elegir si quiere activar o desactiar el sonido cuando se falle una palabra.
- Para poder activar o desactivar el sonido en el error, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición entre activar el sonido o desactivarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF11 Activar o desactivar apoyo visual.

- El usuario podrá elegir si quiere activar o desactiar el apoyo visual para la gestión de errores.
- Para poder activar o desactivar el apoyo visual, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición entre activar apoyo visual o desactivarlo.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF12 Abrir o cerrar fases.

- El usuario podrá establecer las diferentes fases del juego entre abiertas o cerradas.
- En el caso de fases abiertas, el usuario podrá realizar cualquiera de las fases disponibles independientemente del resultado de las fases anteriores. En caso de estar cerradas, el usuario podrá realizar una fase siempre que la anterior haya sido completada.
- En el caso de las fases cerradas, la primera fase siempre estará disponible.
- Para poder abrir o cerrar las fases, se deberá realizar desde el menú de configuración, pulsando el icono correspondiente y seleccionando la opición entre fases abiertas o cerradas.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF13 Establecer o quitar carga visual.

- El usuario podrá añadir o quitar carga visual para mostrar elementos gráficos en el desarrollo de los niveles.
- Para poder establecer o no la carga visual, se deberá realizar desde el menú de configuración pulsando el icono correspondiente y seleccionando la opción entre añadir carga visual o quitarla.
- Es necesario tener un usuario creado y acceder al menú de configuración para realizar esta acción.

RF14 Bloquear y desbloquear acciones en el menú principal.

- El usuario podrá bloquear o desbloquear el acceso a el menú de configuración, creación de usuarios, gestión del vocabulario, acceder a la información de la aplicación.
- Al abrir la aplicación por defecto las acciones mencionadas estarán bloqueadas. Para desbloquearlas, el usuario deberá pulsar el icono del candado. A continuación, se mostrará una pregunta al azar contenida en la base de

datos. Si la respuesta es correcta, se desbloquearán las acciones.

RF15 Añadir palabras al vocabulario de un usuario.

- El usuario podrá añadir palabras al vocabulario del usuario.
- Para poder añadir palabras al vocabulario el usuario tendrá que hacerlo desde el menú de gestión del vocabulario accesible desde el menú de inicio o bien desde el menú de configuración.
- Una vez se ha realizado la búsqueda de las palabras y aparezacan los resultados de dicha búsqueda, el usuario añadirá las palabras deseadas pulsando sobre ellas.
- Es necesario tener un usuario creado para añadir palabras a su vocabulario.

RF16 Eliminar palabras del vocabulario de un usuario.

- El usuario podrá eliminar palabras del vocabulario de otro usuario.
- Para poder eliminar palabras del vocabulario, el usuario tendrá que hacerlo desde el menú de gestión del vocabulario accesible desde el menú de inicio o bien desde el menú de configuración.
- En el panel que muestra las palabras actuales del vocabulario del usuario se podrá eliminar la palabra pulsando y manteniendo la pulsación sobre la imagen.
- Para poder eliminar palabras es necesario tener un usuario creado y no se podrán eliminar palabras si el usuario no tiene más de tres en su vocabulario. Si hay más de tres palabras, solo se podrán eliminar las que no hayan sido trabajadas en ningún ejercicio.

RF17 Cambiar el orden de las palabras del vocabulario.

- El usuario podrá cambiar el orden de las palabras en el vocabulario de un usuario selecccionado.
- Para poder modificar el orden, el usuario deberá acceder al menu de gestión del vocabulario a través del icono en el menú principal o el menú de configuración. En la columna donde aparece el vocabulario del usuario se puede cambiar el orden pulsando la palabra y arrastrnándola a la nueva posición deseada.
- Las tres primeras palabras y todas las palabras que hayan sido trabajadas en alguna de las fases del juego no podrán ser reordenadas.

RF18 Buscar palabras en ARASAAC.

- El usuario podrá buscar en ARASAAC pictogramas para trabajar esas palabras.
- Para poder buscar palabras, el usuario tendrá que acceder al menú de gestión del vocabulario. En el panel de la izquierda se encuentra un campo de texto en el que se indicará la palabra a buscar. Una vez escrita, el usuario tendrá que confirmar el envío y se realizará una búsqueda en ARASAAC mostrando los resultados en caso de encontrarlos.
- Para poder realizar búsquedas tiene que haber un usuario registrado. No es posible buscar palabras de más de 9 caracteres, si así se hicera la aplicación emitrá un aviso indicando el tamaño máximo de las palabras a buscar.

RF19 Arrastrar elementos con la palabra/sílaba al hueco correspondiente.

- El usuario podrá arrastrar los elementos que contienen o bien una palabra o bien una sílaba desde la ejecución de cualquiera de los niveles.
- Cuando el gesto de arrastrar termina y si se ha hecho en el hueco correcto, la aplicación hará saber al usuario que ha acertado. En caso de error, el elemento volverá a su posición inicial y el se indicará el error.
- Para poder realizar esta acción solo será posible con un usuario registrado, que tenga al menos tres palabras en

su vocabulario y desde cualquiera de los niveles disponibles en el juego.

RF20 Reiniciar subnivel.

- El usuario podrá reiniciar un subnivel al finalizarlo completamente.
- Al finalizar un subnivel, se mostrará un menú con la opción de reiniciar el subnivel.
- El usuario podrá realizar esta acción si ha finalizado completamente el subnivel.

RF21 Avanzar al siguiente subnivel.

- El usuario podrá avanzar al siguiente subnivel cuando termine el subnivel previo.
- Al finalizar un subnivel, se mostrará un menú con la opción de avanzar al siguiente subnivel.
- El usuario podrá realizar esta acción si ha finalizado completamente el subnivel y además ha cumplido un mínimo de aciertos en él.

RF22 Almacenamiento de datos tras finalizar un subnivel.

- Cuando un usuario termina un subnivel, el sistema guarda por cada palabra trabajada en cada uno de los diferentes ejercicios los errores y aciertos.

3.1.2. Requisitos no funcionales

Los requisitos no funcionales nos dicen cómo se van a hacer las cosas dentro de la aplicación, cómo va a ser su funcionamiento y nos van a servir para determinar la calidad del sistema.

RNF1 La aplicación debe estar disponible en todo momento independientemente de la conexión del usuario. Por ello solo será necesario tener conexión a internet para añadir palabras al vocabulario.

RNF2 Se deberá optimizar la ocupación en memoria de la aplicación, por ello se deberán eliminar las imágenes de las palabras que dejen de usarse.

RNF3 Los elementos de la interfaz del usuario deben ser reconocidos con facilidad así como deben conocerse las acciones disponibles para cada uno de estos elementos.

RNF4 El usuario deberá ser capaz de navegar entre las diferentes pantallas de la aplicación de forma ágil.

RNF5 Los elementos que aparecen en la pantalla al realizar cualquiera de los ejercicios debe mostrarse claramente en la aplicación.

RNF6 La aplicación debe recuperarse de los errores sin pérdida de información.

RNF7 El sistema debe funcionar con total normalidad para un mínimo de 30 usuarios cada uno de ellos con su propio vocabulario.

RNF8 Los accesos a la base de datos para cargar la información necesaria para el desarrollo de la aplicación no debe ser superior a 5 segundos.

RNF9 El sistema deberá ser fácil de usar para cualquier usuario que utilice la aplicación.

RNF10 El sistema debe proporcionar mensajes de error que sean informativos y útiles para el usuario.

3.2. Diseño de la aplicación

En esta sección se desarrolla todo lo relacionado con el diseño de la aplicación, la arquitectura a utilizar, la base de datos con sus relaciones, las partes más concretas de la lógica del sistema y el diseño de la interfaz de usuario.

3.2.1. Diseño de la base de datos

Para el almacenamiento de la información se ha obtado por una base de datos relacional. La base de datos es local para cada dispositivo y cuenta con varias tablas que vamos a describir a continuación y que se pueden ver en la figura 3.1.

- **User:** tabla que guarda la información de los usuarios. En ella se almacena un ID que identifica al usuario y un nombre que debe ser único.
- **Word**: en esta tabla se almacenan todas las palabras que se añaden en la aplicación. En ella se almacena la ruta local del dispositio donde se almacena la imagen asociada a la palabra, las sílabas que componen la palabra y la propia palabra.
- **Vocabulary:** tabla que relaciona un usuario con una palabra, de esta manera podemos obtener fácilmente el vocabulario de un usuario concreto.
- **Level:** tabla que almacena los diferentes niveles de la aplicación. Se almacena el nombre y la descripción de los niveles. Se pensó así para asegurar que en un futuro se puedan añadir nuevos niveles de forma ágil.
- **Sublevel:** tabla que almacena los diferentes subniveles. En ella también se almacena el nombre y la descripción.
- **Execution:** en esta tabla se almacenan las ejecuciones de un subnivel por parte de un usuario, así como el grupo de palabras al que pertenece y el nivel del que se parte. También se guarda el estado para saber si no se ha realizado nunca, si ha superado el nivel o si debería repetirlo en caso de no obtener suficientes aciertos.
- **SublevelInfo:** esta tabla se encarga de almacenar los errores y aciertos a nivel de palabra para conseguir un nivel de detalle mayor.
- **WordGroup:** encargada de almacenar las palabras que conforman los grupos de trabajo. Éstas se almacenan de tres en tres.
- **Question:** esta tabla almacena las preguntas con sus respuestas correctas que se deben responder para desbloquear las opciones de configuración.
- **Settings:** tabla que almacena todas las opciones disponibles de configuración para cada uno de los usuarios.

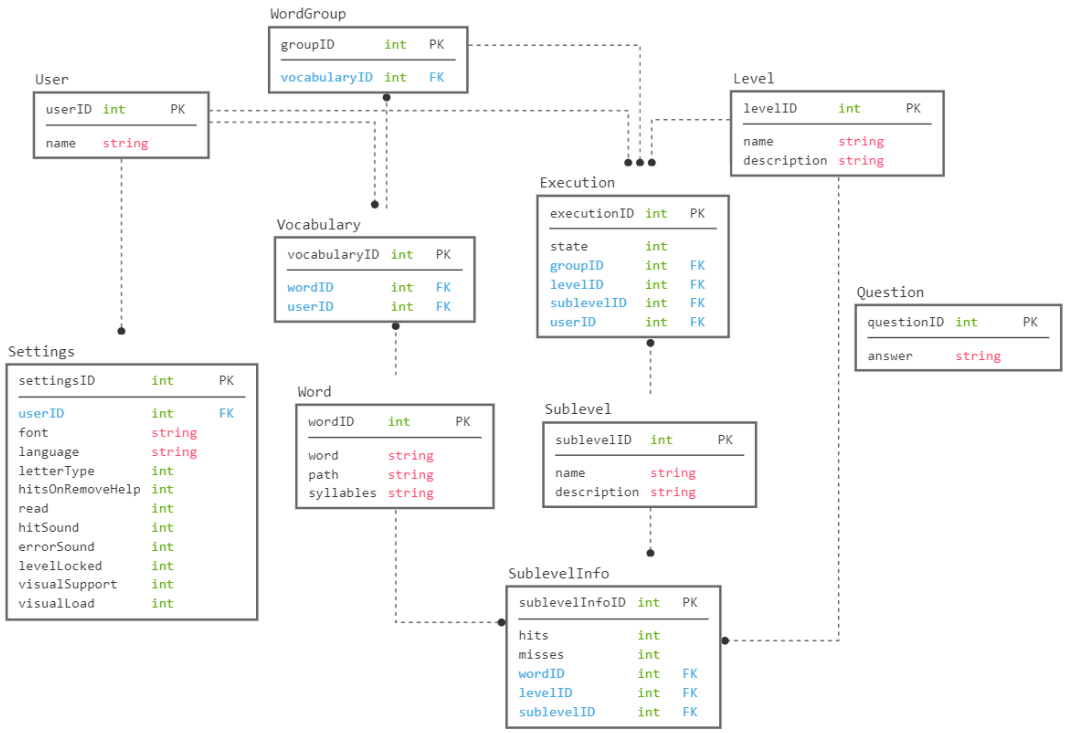

Figura 3.1: Estructura y relaciones de la base de datos.

3.2.2. Lógica del sistema

En este apartado se van describir los diferentes componentes que deberán ser implementados para cumplir con los requisitos funcionales descritos en la sección 3.1.1

Implementación del método de lectura global

En primer lugar definimos los diferentes elementos que forman parte de los niveles, independientemente del tipo de nivel. Unity permite crear elementos llamados prefabs, estos son elementos que se pueden crear y añadir los componentes necesarios. Una vez creados, estos pueden instanciarse en cualquier momento para :

- **Grupos de palabras:** para comenzar un nivel el sistema agrupara el vocabulario del usuario en grupos de palabras formados por tres palabras.
- **Creación de niveles:** una vez creados los grupos de palabras, para cada uno de ellos se deberán crear los tres niveles.
- **Carta:** este componente representa una parte de los elementos que se tienen que relacionar. En cada nivel o subnivel se mostrará como mínimo una carta y como máximo tres. Estas cartas representan las palabras de un grupo de palabras. Contienen la imagen asociada a la palabra, la palabra para el apoyo visual en caso de estar disponible, y el hueco donde se arrastra la palabra o sílaba de la palabra que se está trabajando.
- **Elemento arrastrabale:** El segundo elemento de la asocicación está formado por la palbra que se tiene que arrastra al hueco que está dentro de la carta. Este elemento puede ser una palabra completa o una sílaba en función del nivel en el que nos encontremos. Si el elemento se corresponde con el valor que espera recibir el

hueco de la carta, se producirá un acierto parcial. Si el usuario acierta parcialmente en todos los huecos de la carta, se comunicará el acierto al usuario y avanzaremos a la siguiente pantalla.

La aplicación a su vez se compone de diferentes niveles, cada uno con sus diferentes subniveles.

Los diferentes niveles que podemos encontrar en la aplicación se describen a continuación:

Palabra a imagen con palabra: aparecen en pantalla de una a tres cartas compuestas por una imagen, el nombre de la palabra que se asocia con la imagen y un hueco al que arrastrar el elemento correspondiente. En la parte inferior se encuentra la palabra que hay que arrastrar en el hueco de la imagen correspondiente.

Dentro de este nivel encontramos los siguientes subniveles:

- **Aprendizaje:** Cada palabra aparece de manera individual para que el usuario se familiarice con la palabra y puede asociarla con su imagen.
- **Discriminación:** En este subnivel se trabaja la discriminación de las palabras. Aparecen dos cartas correspondientes a palabras diferentes. Estas palabras siempre pertenecen al mismo grupo de palabras. El usuario tiene que asociar la palabra con la imagen correspondiente.
- **Discriminación inversa:** En este subnivel aumenta la dificultad ya que el usuario encontrará dos palabras para una sola imagen.

Palabra a imagen sin palabra: Este nivel es muy parecido al nivel anterior, la única diferencia es que en este caso la palabra no aparecerá bajo la imagen. Los subniveles funcionan de forma idéntica en este caso teniendo en cuenta la diferencia comentada anteriormente.

Fragmentación silábica: En este nivel se trabajará la lectura silábica. Las palabras que se han trabajado a lo largo de las fases anteriores aparecerán ahora fragmentadas en sílabas. Las cartas ahora tendrán un hueco tanto para la palabra como para cada una de las sílabas que forman la palabra.

En este nivel, los subniveles varían con respecto a los niveles vistos anteriormente.

- **Aprendizaje:** El usuario comenzará a familiarizarse con las sílabas de la palabra en este subnivel. Aparecerá una carta con huecos disponibles para la palabra y para las sílabas.
- **Profundización:** El usuario deberá construir la palabra con las sílabas sin ayuda, ya no se mostrará la palabra completa aumentando así la dificultad.
- **Discriminación:** En este último subnivel el usuario deberá formar la palabra con las sílabas discriminando las sílabas que no pertenecen a la palabra.

3.2.3. Diseño de la interfaz de usuario

En el diseño de la iterfaz de usuario se ha intentado mantener el estilo que podemos encontrar en la versión de Leo con Lula para dispositivos iOS.

El estilo debe ser sencillo y cómodo para el usuario, esto se cumple en la versión actual y por lo

tanto no hay razón para hacer cambios significativos. Por este motivo no se van a encontrar maquetas a lo largo de esta sección.

En la figura 3.2 podemos observar el menú principal de la aplicación. En la imagen se puede apreciar que el usuario ha desbloqueado las opciones de configuración, por lo tanto tiene habilitados los botónes para añadir usuario, gestionar el vocabulario del usuario seleccionado, acceder al menú de configuración, ver la información de la aplicación y obtener ayuda.

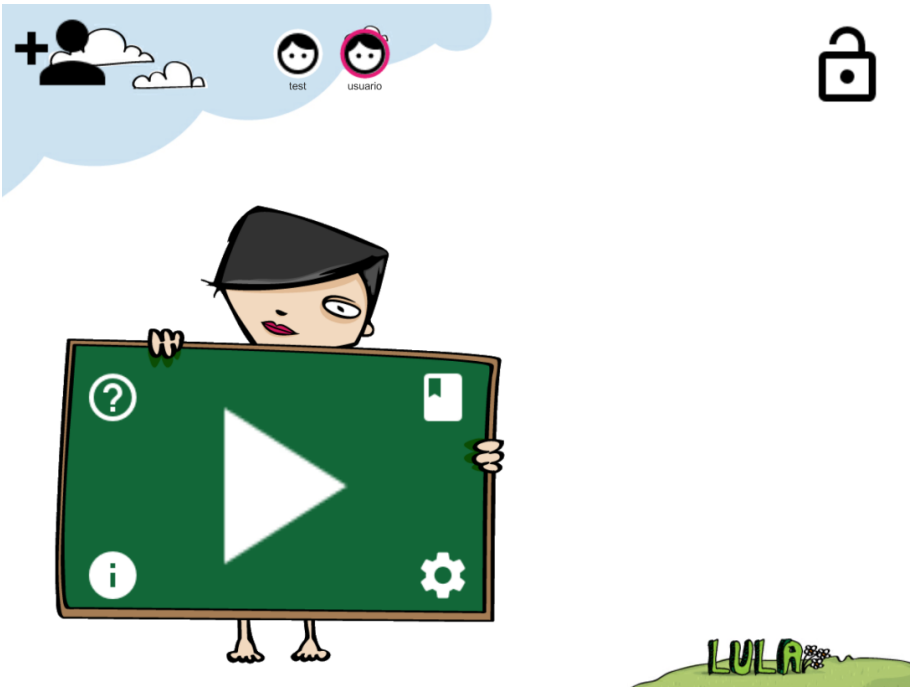

Figura 3.2: Menú principal de la aplicación.

El menú de gestión del vocabulario cuenta con dos bloques principales. En la figura 3.3 se muestran ambos bloques. El primero cuenta con un campo de texto en el que se introducirá la palabra a buscar. El resultado de la búsqueda en ARASAAC se mostrará en el bloque que se encuentra debajo del campo de texto. En la parte de la derecha se mostrará el vocabulario del usuario. Con pulsar en el elemento que aparece en la parte izquierda, éste se añadirá al vocabulario del usuario. Se puede modificar el orden y eliminar las palabras del vocabulario siempre y cuando haya un mínimo de tres palabras o las palabras que se quieran eliminar no hayan sido trabajadas previamente.

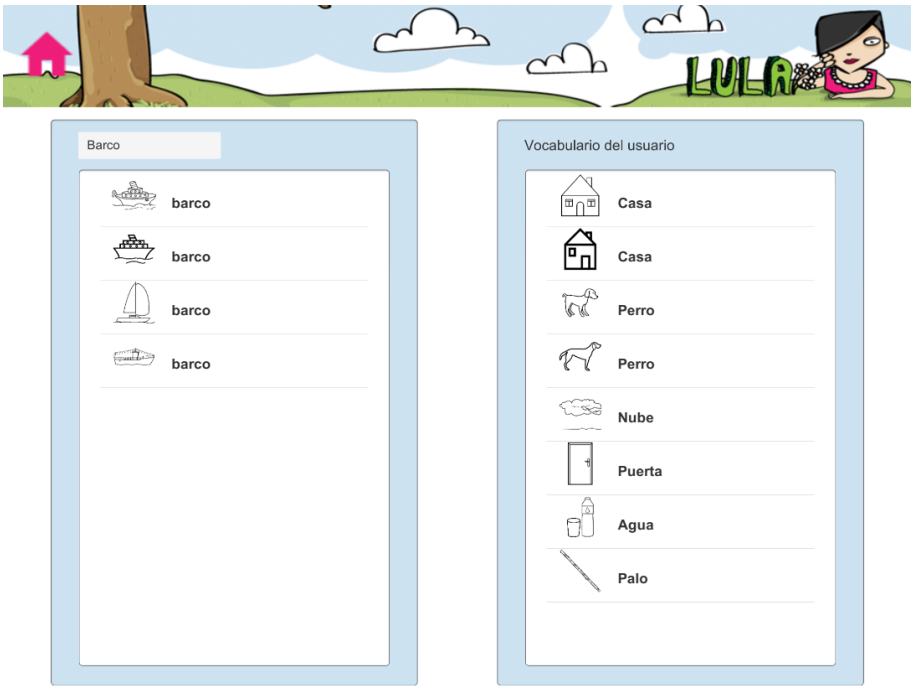

Figura 3.3: Menú de gestión del vocabulario.

Este menú cuenta con algunos cambios con respecto a la versión actual. Se ha eleminiado una ventana de previsualización y edición de la palabra añadida. En él se podía cambiar la fragmentación silábica de la palbra. La existencia de este elemento en la versión actual de Leo con Lula se basa en la corrección de errores a la hora de dividir la palbra en sílabas ya que en algunos casos era necesario realizarlo de forma manual. En esta versión se ha mejorado el algoritmo del slibeador haciendo que esta ventana carezca de utilidad.

El menú de configuración de la aplicación se compone de doce botones y un slider. Estos botones, en el orden en el que aparecen en la figura 3.4 son los siguientes:

- **Cambio de fuente:** es posible cambiar el tipo de fuente entre Helvetica, Sarakanda y Chalkboard.
- **Ayuda de lectura:** se puede activar la lectura de palabras o silábas cuando hay un acierto.
- **Número de aciertos tras los que se elimina la ayuda visual:** se puede configurar cuántos aciertos son necesarios para aumentar la dificultad eliminando la palabra bajo la imagen.
- **Alternar entre mayúsculas y minúsculas:** se puede elegir cómo aparecen las palabras en los ejercicios.
- **Seleccionar el idioma de búsqueda de palabras:** se puede elegir entre 10 idiomas disponibles en ARASAAC para realizar la búsqueda de vocabulario.
- **Activar o desactivar el sonido en el acierto:** es posible configurar si se quiere que la aplicación emita un sonido cuando se acierte una palabra.
- **Activar o desactivar las ayudas visuales:** Se puede configurar que aparezcan ayudas visuales para detectar cuando se ha fallado una palabra.
- **Activar o desactivar el sonido en el error:** es posible configurar si se quiere que la aplicación emita un sonido cuando se falle una palabra.
- **Abrir o cerrar las fases:** con esta opción es posible cerrar o abrir las fases. Si están abiertas es posible realizar ejercicios sin completar subnivles previos.
- **Activar o desactivar la carga visual:** cuando activamos la carga visual podemos ver elemntos gráficos decorativos durante los ejercicios.
- **El slider marca el número de ejercicios por cada nivel:** podemos configurar el número de ejercicios en cada nivel entre 12 y 42.
- **Acceder al vocabulario:** desde esta pantalla podemos acceder a la gestión del vocabulario del usuario.
- **Eliminar usuario:** es posible eliminar el usuario seleccionado desde esta menú.

Figura 3.4: En este menú aparecen las opciones de configuración disponibles en la aplicación.

El menú de selección de nivel tiene como componente principal un menú con pestañas para seleccionar los diferentes niveles. En cada uno de los niveles se muestran los grupos de palabras que tiene el usuario, siempre de tres en tres y en cada grupo de palabra podemos acceder a los diferentes subniveles, explicados en la sección 3.2.2. En la figura 3.5 se muestra seleccionado el primer nivel y en él se observan dos grupos de palabras. Como se puede ver, el resto de subniveles quedan bloqueados y se desbloquean a medida que se completan los subniveles anteriores.

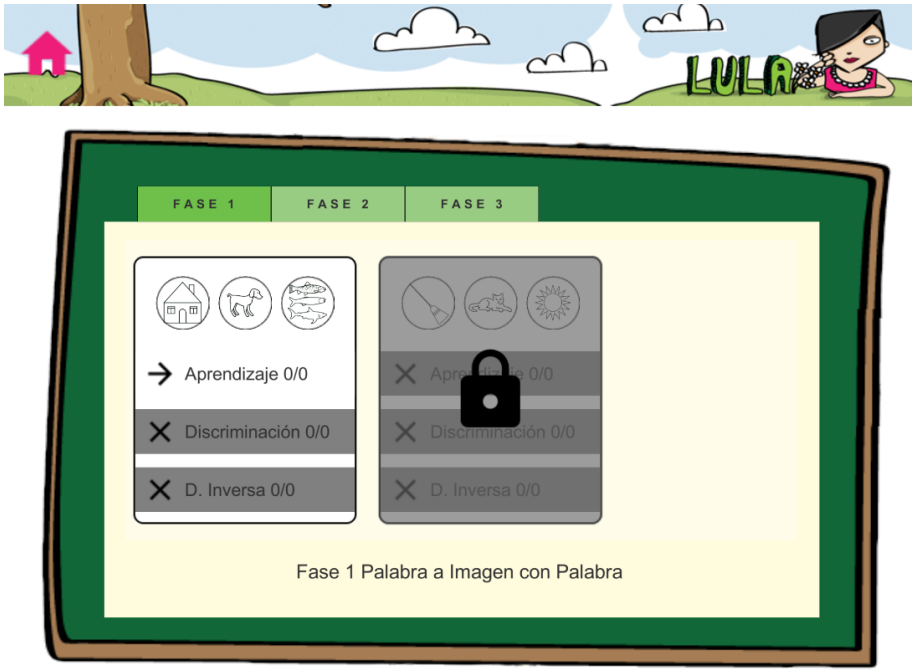

Figura 3.5: Menú de selección de niveles.

Es posible seleccionar los tres niveles disponibles en el juego pulsando en cada una de las pestañas. Una vez seleccionado se mostrarán los subniveles que pertecen a ese nivel con las estadísticas de las ejecuciones realizadas hasta el momento.

En lo que respecta a la ejecución de los diferentes subniveles, observamos cómo en la figura 3.6 se trabaja la palabra 'peces'. En concreto se está trabjando la fragmentación silábica del tercer nivel en el que el usuario debe relacionar la palabra y las sílabas con la imagen. Encima de la carta podemos ver la barra de progreso que muestra cuántos ejercicios hay en este subnivel y qué ejercicio se está realizando. Esta pantalla es muy similar en el resto de niveles y subniveles cambiando únicamente los elementos a arrastrar y el número de cartas que aparecen en pantalla.

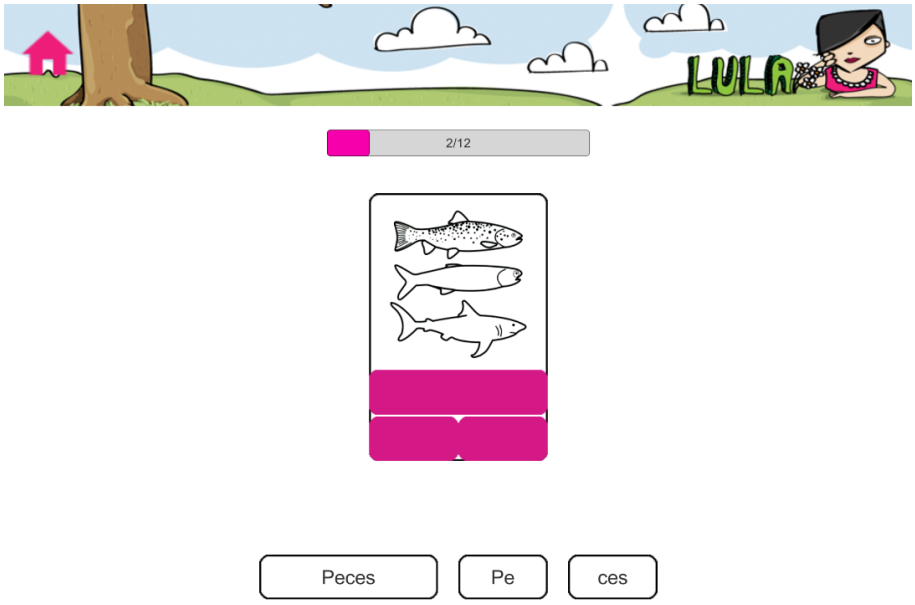

Figura 3.6: Ejemplo de ejecución de uno de los niveles.

4

Implementación

Una vez terminada la fase de análisis y diseño de la aplicación podemos trasladar estas ideas al código y comenzar así con la fase de implementación. En esta fase se desarrollarán las diferentes escenas con las que trabaja Unity, los diferentes módulos para la gestión de la base de datos, comunicación con la API del Centro Aragonés para la Comunicación Aumentativa y Alternativa (ARASAAC) [12] y por supuesto la interfaz y su funcionalidad asociada.

El juego está dividido en varias escenas. Una escena está compuesta por uno o varios GameObject, que es como se denominan a los elementos que componen el juego. Estos objetos por sí mismos no realizan ninguna función por ello es necesario asignar 'Components' a la hora de implementar funcionalidad. En Unity encontramos componentes básicos como por ejemplo el componente Light para dotar a un objeto de luz propia o el componente 'Camera' para crear un punto de vista desde ese objeto. Además es posible añadir Scripts propios a los GameObject que funcionan de la misma forma que un componente.

Los scripts utilizan el funcionamiento interno de Unity al implementar la clase básica 'MonoBehaviour'. Cuando se crea un script que implementa esta clase, se tienen en cuenta dos funciones. La primera de ellas, la función **Start** es llamada antes de que comience a ejecutarse el juego, siendo esta función el lugar más común para realizar inicializaciones. Por otra parte la función **Update** se ejecuta en cada frame en el que esté presente el GameObject, por lo que es útil para añadir movimiento o responder a eventos de entrada del usuario.

4.1. Escenas

Las escenas contienen los menús y entornos dentro de un juego. Una escena podría traducirse en un nivel diferente del juego. Cuando se crea una escena esta cuenta con una Cámara, un componente que crea una imagen de la escena a partir de un punto de vista concreto.

Para esta aplicación se ha decidido crear únicamente tres escenas para simplificar la estructura. Estas escenas son 'Menu', 'Level', 'Vocabulary' y se verán con más profundidad a continuación.

4.1.1. Escena del menú principal

Dentro de esta escena se van a mostrar diferentes 'Canvas'. Un Canvas es un elemento que nos ofrece Unity donde se van a agrupar todos los elementos de la interfaz de usuario. En este caso contaremos con tres canvas.

- StarterCanvas: Este elemento se mostrará nada más iniciar la aplicación y actuará como pantalla de carga inicial. Cuenta con una animación de desvanecimiento y cuando esta termina este cavnas queda deshabilidado y se muestra en su lugar el canvas del menú principal.
- MainCanvas: Este canvas contiene los elementos del menú principal que son los diferentes botones que se muestran en la figura 3.2 agrupados en un mismo objeto.
- SettingsCanvas: Si pulsamos el botón con forma de engranaje que aparece en el menú principal este nos lleva al menú de configuración. No se trata de otra escena, lo que hacemos es ocultar MainCanvas para mostrar SettingsCanvas. Del mismo modo, pulsando en el icono con forma de casa que aparece en la parte superior izquierda de la figura 3.4 haremos el proceso inverso para volver a mostrar el menú principal.

4.1.2. Escena de gestión del vocabulario

Esta escena fue añadida ya que, por cómo fucniona Unity, y la funcionalidad necesaria en esta parte, separarlo del menú principal mejoraría el rendimiento. En esta escena tenemos dos contenedores, uno de ellos para almacenar las palabras que resulten de la búsqueda y el segundo es para mostrar las palabras que el usuario tiene en su vocabulario.

Ambos elementos pertececen a la librería UnityUIExtensions [1] en la cual podemos encontrar elementos de interfaz gráfica que no aparecen en los paquetes por defecto de Unity. Estos dos elementos son una lista que implementa la funcionalidad para hacer scroll en la que se almacenan los resultados de la búsqueda de palabras y el segundo, el que cuenta con el vocabulario, es una lista reordenable que nos permite cambiar el orden de las palabras en el vocabulario del usuario.

4.1.3. Escena de gestión de niveles y subniveles

Esta escena se muestra cuando el usuario pulsa el botón para jugar en el menú principal. Como se muestra en la figura 3.5 el usuario podrá seleccionar el nivel o fase y dentro de cada una de ellas y por cada grupo de palabras, el subnivel a practicar siempre que se encuentre desbloqueado.

Esta escena cuenta con un solo canvas que a su vez tiene varios GameObjects para agrupar los elementos a mostrar en cada momento. Por una parte, la pizarra que se muestra y menú de pestañas para elegir el subnivel y por otra parte el lugar donde se van a colocar las cartas con las imágenes y los huecos para las palabras o sílabas y los elementos arrastrables a estos huecos.

[1] UnityUIExtensions https://bitbucket.org/UnityUIExtensions/unity-ui-extensions/wiki/Home

En cuanto el usuario selecciona un subnivel disponible, el programa oculta este menú y muestra el primer ejercicio a realizar por el usuario.

4.2. Drag & drop

La mecánica principal de Leo con Lula se basa en la posibilidad de arrastrar elementos para relacionar palabras o sílabas con imágenes. Esto es conocido como 'Drag & drop' y en esta sección se muestra cómo ha sido implementado.

Para ello se crean dos scripts, 'DragDrop.cs' e 'ItemSlot.cs'. Ambos heredan de la clase básica 'MonoBehaviour' de la que derivan todos los scripts que se implementan en Unity. El script 'DragDrop.cs' implementa las interfaces 'IPointerDownHandler', 'IBeginDragHandler', 'IEndDragHandler' e 'IDragHandler' que ofrecen funciones para implementar el gesto que permita arrastrar un objeto. Por su parte, 'ItemSlot.cs' implementa la interfaz 'IDropHandler' que permite realizar acciones cuando un objeto sea soltado en este hueco.

El proceso se describe a continuación:

- Cuando un objeto que tiene como componente el script 'DragDrop.cs' es pulsado y se mantiene esta pulsación, comienza el gesto conocido como 'drag'. Cuando esto pasa, se llama al callback 'OnBeginDrag()' que en este caso se encarga simplemente de deshabilitar la interacción con este elemento y aplicar un efecto visual para que el usuario sea consciente de que ha comenzado el gesto.
- En el momento en el que el usuario que mantiene la pulsación sobre el objeto comienza a arrastrarlo a una nueva posición, es cuando se llama al callback 'OnDrag()' que únicamente actualiza la posición del objeto que es arrastrado a la posición donde el usuario mantiene la pulsación, vease el listado 4.1.

Código 4.1: Esta función se ejecuta cuando se arrastra el objeto.

```
28      public void OnDrag(PointerEventData eventData)
29      {
30          rect.anchoredPosition += eventData.delta / canvas.scaleFactor;
31      }
```

- Cuando el script 'ItemSlot.cs' es asignado como componente a un objeto, también se le asigna el valor esperado en el hueco. Es decir, si queremos trabajar la palabra 'Casa', porque está en el vocabulario del usuario, en la instancia de 'ItemSlot.cs' el valor esperado sera 'Casa'. Por lo tanto, cuando el usario arrastra el elemento y lo suelta sobre el hueco disponible, se ejecuta el callback 'OnDrop()', reconociendo que 'algo' ha sido soltado. Esta función ha sido codificada para que compruebe si el valor del objeto depositado en el hueco es el mismo que el valor esperado. Si es así como se puede ver en el listado 4.2, se comunicará el acierto al GameManager y se marcará esta acción como un acierto.

Código 4.2: Esta función se ejecuta cuando un objeto es arrastrado hacia otro que tiene como componente el script ItemSlot.

```
11      public void OnDrop(PointerEventData eventData)
12      {
13          DragDrop drag = eventData.pointerDrag.GetComponent<DragDrop>();
14          Debug.Log("OnDrop");
15          if (eventData.pointerDrag != null)
16          {
17              eventData.pointerDrag.GetComponent<RectTransform>().position =
                    GetComponent<RectTransform>().GetChild(0).GetComponent<RectTransform>().position;
18          }
19          if (drag.value == value)
20          {
21              drag.correct = true;
22              filled = true;
23              GameManager.instance.Hit();
24          }
25          else
26          {
27              GameManager.instance.Miss();
28          }
29      }
```

En este punto el movimiento no estaría completado. 'ItemSlot.cs' habría detectado que se ha soltado un objeto y se habría comprobado el fallo o acierto al soltarlo, pero esa información tiene que ser procesada por el 'GameManager'.

- Por último, cuando el objeto es soltado, en el script 'DragDrop.cs' se ejecutará el callback 'OnEndDrag()' indicando la finalización del movmiento. En este punto, sabremos si se ha depoistado en el hueco correcto. Si es así, como se puede observar en el listado 4.3, el objeto quedará bloqueado y no podrá volver a moverse indicando así el acierto. En caso de error, como se puede ver en la línea 40 en el listado 4.3 el objeto volvería a la posción incicial.

Código 4.3: Esta función se ejecuta cuando acaba el gesto de arrastrar un objeto.

```
33      public void OnEndDrag(PointerEventData eventData)
34      {
35          canvasGroup.alpha = 1f;
36          canvasGroup.blocksRaycasts = true;
37          if (!correct)
38          {
39              canvasGroup.blocksRaycasts = false;
40              Invoke("ResetPosition", 1.5f);
41          }
42          else
43          {
44              canvasGroup.blocksRaycasts = false;
45          }
46      }
```

4.3. Comunicación con la API de ARASAAC

ARASAAC es un Sistema Aumentativo y Alternativo de Comunicación que está basado en el uso de pictogramas para facilitar la comunicación a personas con necesidades especiales. En su portal almacenan un catálogo de pictogramas que son usados en Leo con Lula para el método de lectura global.

Puesto que es una parte importante para la aplicación, se ha creado un módulo de comunicación con la API para agilizar la descarga de pictogramas.

La API ofrece una forma directa de realizar una búsqueda de palabras pero para gestionar el error al arrancar la aplicación se obtienen todas las palabras disponibles en la plataforma. La función 'GetKeywords' se encarga de esto y se puede ver en la figura listado 4.4

Como se aprecia en la figura, a la hora de realizar peticiones utilizamos el objeto UnityWebRequest. Este objeto maneja el flujo de la comunicación HTTP con un servicio web. Utilizando el método estático 'Get', se crea la petición HTTP y esta se envía quedando a la espera de respuesta mediante el método 'SendWebRequest()'.

Una vez recibida la respuesta, comprobamos los posibles errores al realizar la petición, si todo ha ido correctamente se procesa la respuesta para obtener la lista de palabras disponible.

Código 4.4: Esta función obtiene las palabras de los pictogramas disponibles en ARASAAC.

```
52      private IEnumerator GetKeywords()
53      {
54          string url = String.Format("https://api.arasaac.org/api/keywords/{0}", settings.search_language);
55          var request = UnityWebRequest.Get(url);
56
57          yield return request.SendWebRequest();
58
59          if (request.isNetworkError || request.isHttpError)
60          {
61              Debug.Log(request.error);
62          }
63          else
64          {
65              JsonValue jsonArray = JsonValue.Parse(request.downloadHandler.text);
66              List<string> words = new List<string>();
67              JsonValue jsonWords = jsonArray["words"];
68              foreach (JsonPrimitive child in jsonWords)
69              {
70                  words.Add(child);
71              }
72              keywords = words.AsQueryable<string>();
73          }
74      }
```

Explicado el funcionamiento de las peticiones HTTP, la descarga de pictogramas funciona de forma similar y en este caso vamos a centrarnos en el proceso de actualización de la interfaz con el resultado de las búsquedas.

En primer lugar, esta funcionalidad se encuentra en la función 'Update()'. Esta función está disponible en todos los objetos de tipo 'MonoBehaviour' y se ejecuta en cada frame si el objeto está presente en la escena. Aunque la función se ejecute cada frame, solo necesitamos que realice búsquedas cuando el usuario confirme la palabra a buscar. Como se observa en el listado 4.5, condicionamos la búsqueda a que exista un teclado instanciado y sea visible. Si además de eso el usuario confirma la búsqueda, se realizan comprobaciones sobre la palabra a buscar.

Código 4.5: Condiciones para ejecución de la búsqueda en ARASAAC

```
76      void Update()
77      {
78          if ((TouchScreenKeyboard.visible == false && keyboard != null))
79          {
80              if (keyboard.status == TouchScreenKeyboard.Status.Done ||
81                  Input.GetKeyDown(KeyCode.Return))
82              {
```

La palabra, ahora sí, se busca en la lista de palabras disponibles en ARASAAC para posteriormente

realizar la petición al servidor para descargar el pitograma asociado. La descarga del pictograma se realiza mediante una petición HTTP como hemos explicado anteriormente. Una vez descargados los pictogramas para las palabras que cumplan la búsqueda, se instancian elementos visuales como los que aparecen en la figura 3.3 para que el usuario pueda añadir las palabras a su vocabulario.

4.4. Base de datos

Se han creado diferentes módulos para facilitar la comunicación con la base de datos. Para ello se utilizará el sistema de SQLite y la librería 'Mono.Data.SQlite'. Siguiendo el esquema que se muestra en la sección 3.2.1 se han creado diferentes módulos para cada una de las tablas. Para hacer este módulo extensible, se ha creado una clase genérica de la que van a partir cada una de las tablas.

Esta clase se llama 'SQLiteHelper' y va a facilitar las conexiones con la base de datos en todo momento y va a permitir gestionar las consultas.

En primer lugar se define la conexión a la base datos, el nombre de la base de datos y la cadena de conexión.

Código 4.6: Estructura de la clase abstracta de la base de datos

```
9  namespace Repository
10 {
11     public class SqliteHelper
12     {
13         private const string Tag = "Riz: SqliteHelper:\t";
14
15         private const string database_name = "luladb.sqlite";
16
17         public string db_connection_string;
18         public IDbConnection db_connection;
19
20         public SqliteHelper()
21         {
22             db_connection_string = "URI=file:" + Application.persistentDataPath
23                 + "/" + database_name + ";foreign keys=true;";
24             db_connection = new SqliteConnection(db_connection_string);
25             db_connection.Open();
26             IDbCommand dbcmd = getDbCommand();
27             dbcmd.CommandText = "PRAGMA foreign_keys = ON;";
28             dbcmd.ExecuteNonQuery();
29         }
```

En esta clase se definen las funciones virtuales que serán implementadas por las clases que hereden de la clase principal, vease el listado 4.7.

Código 4.7: Definición de las funciones virtuales para la base de datos

```
36          public virtual IDataReader getDataById(int id)
37          {
38              Debug.Log(Tag + "This_function_is_not_implemented");
39              throw null;
40          }
41
42          public virtual IDataReader getDataByString(string str)
43          {
44              Debug.Log(Tag + "This_function_is_not_implemented");
45              throw null;
46          }
47
48          public virtual void deleteDataById(int id)
49          {
50              Debug.Log(Tag + "This_function_is_not_implemented");
51              throw null;
52          }
53
54          public virtual void deleteDataByString(string str)
55          {
56              Debug.Log(Tag + "This_function_is_not_implemented");
57              throw null;
58          }
59
60          public virtual IDataReader getAllData()
61          {
62              Debug.Log(Tag + "This_function_is_not_implemented");
63              throw null;
64          }
65
66          public virtual void deleteAllData()
67          {
68              Debug.Log(Tag + "This_function_is_not_implemnted");
69              throw null;
70          }
71
72          public virtual IDataReader getNumOfRows()
73          {
74              Debug.Log(Tag + "This_function_is_not_implemnted");
75              throw null;
76          }
```

Por último, se añaden funciones de utilidad comunes al resto de tablas con funcionalidad básica como por ejemplo obtener todos los datos de una tabla, eliminar todo el contido, obtener el número de filas. La función 'getDbCommand()' que aparece en el listado 4.8 será útil para crear cualquier comando SQL que ejecutemos.

Código 4.8: Definición de funciones genéricas en la base de datos

```
79          public IDbCommand getDbCommand()
80          {
81              return db_connection.CreateCommand();
82          }
83
84          public IDataReader getAllData(string table_name)
85          {
86              IDbCommand dbcmd = db_connection.CreateCommand();
87              dbcmd.CommandText = "SELECT * FROM " + table_name;
88              IDataReader reader = dbcmd.ExecuteReader();
89              Debug.Log(dbcmd.CommandText);
90              return reader;
91          }
92
93          public void deleteAllData(string table_name)
94          {
95              IDbCommand dbcmd = db_connection.CreateCommand();
96              dbcmd.CommandText =
97                  "DROP TABLE IF EXITS " + table_name;
98              dbcmd.ExecuteNonQuery();
99          }
100         public IDataReader getNumOfRows(string table_name)
101         {
102             IDbCommand dbcmd = db_connection.CreateCommand();
103             dbcmd.CommandText = "SELECT COALESCE(MAX(id) + 1, 0) FROM " + table_name;
104             IDataReader reader = dbcmd.ExecuteReader();
105             return reader;
106         }
107         public void close()
108         {
109             db_connection.Close();
110         }
```

Este clase nos permite ahora añadir cualquier tabla y definir sus características de forma sencilla a la vez que nos garantiza flexibilidad en el futuro para cualquier cambio en la base de datos que sea necesario.

4.5. Gestión de los subniveles

Cada uno de los niveles tiene unas reglas concretas, a su vez, cada subnivel funciona de forma diferente. Uno de los objetivos de este trabajo es generalizar la ejecución de los subniveles.

Para implementar esta parte se ha dividido el trabajo en tres módulos: 'GameManager.cs', 'UIManager.cs' y 'GameEvents.cs'.

'GameManager.cs' se encargará de gestionar la lógica de cada uno de los niveles. Cuando el usuario seleccione un nivel para comenzar a jugar, automáticamente este script almacenará la información del subnivel y el nivel al que pertenece este subnivel. Esto se debe a que los elementos que componen un subnivel y su disposición en pantalla varía en cada uno de ellos.

Se utilizará una clase para almacenar la información a mostrar en cada uno de los ejercicios del nivel; esto es, número de pictogramas a mostrar con su información y número de palabras/sílabas que aparecerán en la parte inferior que el usuario deberá arrastrar para completar el ejercicio. Esta clase es 'StageData.cs' y es el script 'GameManager.cs' el encargado de generar toda esta información.

Código 4.9: Esta clase representa los ejercicios a realizar en un subnivel.

```
6  public class StageData
7  {
8      //Los pictogramas a mostrar
9      [SerializeField] private List<WordEntity> options;
10     public List<WordEntity> Options { get { return options; } }
11     //Las palabras/silabas a arrastrar hacia el hueco
12     [SerializeField] private List<string> words;
13     public List<string> Words { get { return words; } }
14
15     public StageData()
16     {
17         options = new List<WordEntity>();
18         words = new List<string>();
19     }
20 }
```

Una vez cargado el nivel, tendremos una lista con la información de los ejercicios. Estos ejercicios irán apareciendo de uno en uno cada vez que el usuario los vaya completando.

Como se aprecia en el listado 4.10 la función 'Accept()' espera un tiempo definido para mostrar el siguiente ejercicio, siempre y cuando el nivel no esté finalizado. La función 'WaitTillNextPicto()' se encarga de de esperar para poder avanzar al siguiente ejercicio.

Código 4.10: Esta clase sirve los ejercicios según se van completando.

```
296     public void Accept()
297     {
298         if (IsFinished)
299         {
300             events.UpdateSlider?.Invoke(currentStage -1, maxExercises);
301             return;
302         }
303         if (IE_WaitTillNextPicto != null)
304         {
305             StopCoroutine(IE_WaitTillNextPicto);
306         }
307         IE_WaitTillNextPicto = WaitTillNextPicto();
308         StartCoroutine(IE_WaitTillNextPicto);
309     }
```

La función 'Display()' se encarga de obtener el siguiente ejercicio de la lista de ejercicios disponibles en el nivel así como invocar las funciones del gestor de eventos que se encargan de actualizar la barra de progreso del nivel y actualizar los elementos en función del siguiente ejercicio.

Código 4.11: Esta función invoca el evento para actualizar la interfaz con el siguiente ejercicio

```
264     void Display()
265     {
266         StageData stageData = GetNextStage();
267         events.UpdateSlider?.Invoke(currentStage -1, maxExercises);
268         events.UpdatePictoUI?.Invoke(stageData, levelType);
269     }
```

Como se ha comprobado en el listado 4.11 se hacen llamadas a una clase que gestiona los eventos del juego. Esta clase se llama 'GameEvents' y como se puede ver en el listado 4.12, en ella se declaran los callbacks necesarios para actualizar los pictogramas de cada ejercicio, actualizar la barra de progreso, mostrar las ayudas visuales en el error y mostrar el menú de finalización del subnivel.

También se ha creado una clase para actualizar únicamente la parte visual. Esta clase cuenta con una estructura como se muestra en el listado 4.13 en la que se asignan todos los elementos de la interfaz que van a ser actualizado.

La clase 'UIManager' cuenta con una instancia de la clase que gestiona los eventos y además cuenta con algunos elementos que se neceistan para crear los ejercicios, como pueden ser las cartas con los pictogramas y los elementos arrastrables para trabajar las palabras. Para poder gestionar los eventos, este script se tiene que suscribir a las funciones que actualizan la interfaz. Como se puede ver en el listado 4.15, cuando este objeto se habilita se suscribe al evento que actualiza tanto la interfaz como la barra de progreso. Y de la misma forma que nos suscribimos a estos métodos, hay que

Código 4.12: Esta clase gestiona los eventos de la partida.

```
7  public class GameEvents : ScriptableObject
8  {
9      public delegate void UpdatePictoUICallback(StageData stage, GameManager.LevelType levelType);
10     public UpdatePictoUICallback UpdatePictoUI;
11
12     public delegate void DisplayEndScreenCallback();
13     public DisplayEndScreenCallback DisplayEndScreen;
14
15     public delegate void UpdatePictoCallback(List<WordEntity> words);
16     public UpdatePictoCallback UpdatePicto;
17
18     public delegate void UpdateSliderCallback(int currentStage, int maxStages);
19     public UpdateSliderCallback UpdateSlider;
20
21     public delegate void DisplayErrorPictoCallback();
22     public DisplayErrorPictoCallback DisplayErrorPicto;
23
24     [HideInInspector]
25     public int hits;
26     [HideInInspector]
27     public int misses;
28 }
```

Código 4.13: Esta estructura contiene los elementos que forman parte de la interfaz de un subnivel.

```
15 public struct UIElements
16 {
17     [SerializeField] RectTransform pictosContentArea;
18     public RectTransform PictosContentArea { get { return pictosContentArea; } }
19     [SerializeField] RectTransform draggableElementsArea;
20     public RectTransform DraggableElementsArea { get { return draggableElementsArea; } }
21     [SerializeField] Slider progressBar;
22     public Slider ProgressBar { get { return progressBar; } }
23     [Space]
24     [SerializeField] CanvasGroup mainCanvasGroup;
25     public CanvasGroup MainCanvasGroup { get { return mainCanvasGroup; } }
26     [SerializeField] RectTransform finishUIElement;
27     public RectTransform FinishUIElement { get { return finishUIElement; } }
28
29     public void UpdateValueProgressBar(int value) { progressBar.value = value; }
30 }
```

desuscribirse cuando se desahabilita el componente.

Código 4.14: Parte de la clase UIManager donde se definen los elementos gráficos y se comunincan con el gestor de eventos.

```
31  public class UIManager : MonoBehaviour
32  {
33      [Header("References")]
34      [SerializeField] GameEvents events;
35
36      [Header("UIElements")]
37      [SerializeField] GameObject defaultCardPrefab;
38      [SerializeField] GameObject cardPrefabSyllables;
39      [Space]
40      [SerializeField] GameObject dragPrefab;
41      [SerializeField] GameObject dragPrefab2Syllables;
42      [SerializeField] GameObject dragPrefab3Syllables;
43      [SerializeField] GameObject dragPrefab4Syllables;
44      [Space]
45      [SerializeField] GameObject slotPrefab1Syllable;
46      [SerializeField] GameObject slotPrefab2Syllables;
47      [SerializeField] GameObject slotPrefab3Syllables;
48      [SerializeField] GameObject slotPrefab4Syllables;
49
50      [SerializeField] UIElements uIElements;
```

Código 4.15: Funciones de suscripción y desuscripción a eventos para la actualización de la interfaz.

```
52      private void OnEnable()
53      {
54          events.UpdatePictoUI += UpdatePictoUI;
55          events.UpdateSlider += UpdateSlider;
56      }
57
58      private void OnDisable()
59      {
60          events.UpdatePictoUI -= UpdatePictoUI;
61          events.UpdateSlider -= UpdateSlider;
62      }
```

Por último, las funciones ‘UpdatePictoUI’ y ‘UpdateSlider’ únicamente actualizan los valores que se muestran en pantalla. En concreto ‘UpdatePictoUI’, como se muestra en el listado 4.16, recibe como parámetros la información del ejercicio a mostrar y el tipo de subnivel para definir el tipo de carta a mostrar puesto que puede ser con un solo hueco para depositar la palabra, o además de ese hueco, los diferentes huecos para cada sílaba.

Código 4.16: Cabecera de la función que actualiza la parte visual del ejercicio.

```
64    void UpdatePictoUI(StageData stage, GameManager.LevelType levelType)
```

5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este TFG se propuso trabajar en el proyecto Leo con Lula, una aplicación disponibe en iOS que cuenta también con versiones en Android pero que están obsoletas. Por ello se propuso crear una nueva versión. Dado el beneficio que supone mantener el código en una única versión multiplataforma, se descartó realizar una versión nativa para Android.

De las opciones disponibles se decidió utilizar Unity por las ventajas que tiene a la hora de crear código nativo, por la cantidad de recursos que se pueden aprovechar tanto ahora como en el futuro de la Unity Asset Store y la capacidad gráfica que nos brinda el motor de juegos que posee.

En el proceso de diseño se hicieron algunos cambios para mejorar la aplicación con respecto a la versión actual en iOS como por ejemplo mejorar la gestión del vocabulario.

Desde el punto de vista de la implementación, además de la funcionalidad básica existente en la versión actual de la aplicación, se ha conseguido generalizar la creación y ejecución de los niveles para que estos sean flexibles y tengan un mejor mantenimiento. Se ha creado una clase para realizar la comunicación con la API de ARASAAC de una forma eficiente. Se ha desarrollado con exito un script que reproduce de forma satisfactoria el gesto de Drag & Drop, fundamental en esta aplicación.

La aplicación es funcional aunque sería necesario realizar pruebas exhaustivas con usuarios reales. Se ha probado en varios dispositivos aunque en iOS no se ha podido compilar la aplicaicón ya que es necesario tener una cuenta de desarrollador para poder instalar aplicaciones propias y es algo que estoy trabajando con mi tutor.

5.2. Trabajo futuro

Como trabajo futuro sería interesante realizar pruebas en diferentes dispositivos y con usuarios reales para conocer posibles errores en la implementación, refinar aún más la funcionalidad para poder obtener un producto que pueda ser publicado y pueda sustituir la versión actual en iOS teniendo

además el mismo producto en dispositivos Android.

También sería interesante realizar un estudio sobre el impacto que puede llegar a tener una aplicación de este estilo en el aprendizaje de los niños con TEA.

Bibliografía

[1] J. Gomez, L. Jaccheri, J. C. Torrado, and G. Montoro, "Leo con lula, introducing global reading methods to children with asd," in *Proceedings of the 17th ACM Conference on Interaction Design and Children*, pp. 420–426, 2018.

[2] "Leo con lula | lectura global para niñ@s con tea." https://leoconlula.com/. (Último acceso 06/07/2020).

[3] "Aprender a leer 1: app leo con grin en ios y android educaplanet apps." https://www.educaplanet.com/educaplanet/juegos/aprender-a-leer-con-grin/. (Último acceso 06/07/2020).

[4] "La aplicación para aprender a leer – yotambienleo.com." https://yotambienleo.com/app/. (Último acceso 06/07/2020).

[5] "Home | gloread." https://evgfinkel.wixsite.com/gloread. (Último acceso 06/07/2020).

[6] "Vocales y números para niños en app store." https://apps.apple.com/ve/app/vocales-y-n%C3%BAmeros-para-ni%C3%B1os/id826439703. (Último acceso 06/07/2020).

[7] "Prolexyco by editorial geu." https://appadvice.com/app/prolexyco/1209460897. (Último acceso 06/07/2020).

[8] "Multiplatform | unity." https://unity.com/features/multiplatform. (Último acceso 06/07/2020).

[9] "React native · a framework for building native apps using react." https://reactnative.dev/. (Último acceso 06/07/2020).

[10] "Ionic - cross-platform mobile app development." https://ionicframework.com/. (Último acceso 06/07/2020).

[11] "Cross-platform with xamarin | .net." https://dotnet.microsoft.com/apps/xamarin/cross-platform. (Último acceso 06/07/2020).

[12] "Arasaac: Centro aragones para la comunicacion aumentativa y alternativa." http://www.arasaac.org/. (Último acceso 06/07/2020).


UAM
UNIVERSIDAD AUTONOMA
DE MADRID